WORDREG: MITIGATING THE GAP BETWEEN TRAINING AND INFERENCE WITH WORST-CASE DROP REGULARIZATION

Jun Xia[1,2], *Ge Wang*[2], *Bozhen Hu*[2], *Cheng Tan*[2], *Jiangbin Zheng*[2], *Yongjie Xu*[2], *Stan Z. Li*[2†]

[1] Zhejiang University, Hangzhou, 310058, China
[2]AI Division, School of Engineering, Westlake University, Hangzhou, 310030, China

ABSTRACT

Dropout has emerged as one of the most frequently used techniques for training deep neural networks (DNNs). Although effective, the sampled sub-model by random dropout during training is inconsistent with the full model (without dropout) during inference. To mitigate this undesirable gap, we propose WordReg, a simple yet effective regularization built on dropout that enforces the consistency between the outputs of different sub-models sampled by dropout. Specifically, WordReg first obtains the worst-case dropout by maximizing the divergence between the outputs with two sub-models with different random dropouts. And then, it encourages the agreements between the outputs of the two sub-models with worst-case divergence. Extensive experiments on diverse DNNs and tasks reveal that WordReg can achieve notable and consistent improvements over non-regularized models and yields some state-of-the-art results. Theoretically, we verify that WordReg can reduce the gap between training and inference. The code for reproducing the results will be released.

Index Terms— Image Recognition, Language Understanding, Graph Mining, Dropout, Regularization

1. INTRODUCTION

Deep Neural Networks (DNNs) have achieved spectacular results in diverse tasks including image recognition, language understanding, etc. However, DNNs often suffer from overfitting and poor generalization. To alleviate these issues, some regularizations [1, 2, 3] have been proposed. Among them, Dropout [1] is the most popular technique for its simplicity and effectiveness. Specifically, it performs implicit ensemble by simply dropping a certain proportion of neurons from the neural networks during training. However, previous works [4, 5] have pointed out that the potential deficiency of Dropout is that it creates the undesirable inconsistency between the training and inference, i.e., the sampled sub-model by random dropout during training is inconsistent with the full model (without dropout) during inference. Therefore, they [4, 5] impose L_2 regularization on the inconsistent hidden states. However, they have not been widely used because: (1) the L_2 distance on hidden states is not in the same space as the main training objective of classification (i.e., maximizing the log-likelihood on the output probability distribution), which will hinder the optimization process; (2) the sub-model consistency should be controlled on the output probability level since the main objective of classification is to make the prediction distribution to be closer to the ground-truth distribution. The smaller hidden states distance cannot guarantee the model outputs to be closer. In contrast, WordReg enforces sub-model consistency on the output probability level using KL divergence.

In this paper, we propose a simple yet effective regularization, dubbed WordReg, to reduce the inconsistency between training and inference. More specifically, for each mini-batch training, we first feed the data into the sub-model sampled with a random dropout. And then, we obtain the other worst-case dropout by maximizing the divergence between the outputs with two sub-models with different dropouts. This process can be formulated as a Binary Quadratic Programming (BQP) problem and we contribute an efficient and effective approach to solve it. Finally, we maximize the agreements between the output probability distributions of the two sub-models that sampled with random dropout and worst-case dropout, respectively. Alternatively, we can also minimize the KL divergence between the output probability distributions of two sub-models that are both sampled with random dropouts. Compared with this regularization, WordReg is task-dependent and possesses a stronger regularization ability, which we verify through extensive experiments in Section 4.4. We highlight our contributions as **(1)** We propose WordReg, a simple yet effective regularization built on dropout, which can be universally applied to diverse neural networks and tasks. **(2)** Theoretically, We explain that WordReg minimizes the upper bound of the inconsistency between training and inference in essence. **(3)** Experiments on image, texts, and graph data reveal that WordReg achieves notable and consistent improvements over non-regularized models and yields some state-of-the-art results.

2. METHODOLOGY

We show the overall framework of WordReg in Figure 1. Formally, considering a sub-model that takes sample $\mathbf{x}$ as input

† Stan Z. Li is the corresponding author.

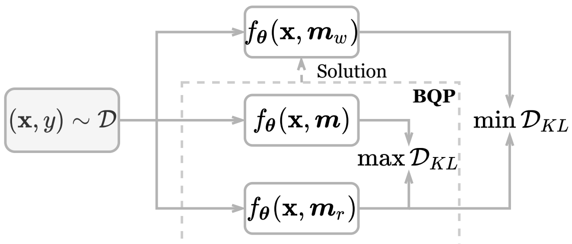

Fig. 1. Illustration of Worst-case drop Regularization (WordReg). We obtain the worst-case drop mask vector $\boldsymbol{m}_w$ (relative to $\boldsymbol{m}_r$) via solving a BQP problem.

and $\boldsymbol{m}$ as a mask vector denoting which neurons of the neural network should be dropped, the output (after softmax) of the sub-model is $f_\theta(\mathbf{x}, \boldsymbol{m})$. For the i-th unit m_i of the mask vector $\boldsymbol{m}$, $m_i = 1$ indicates that the neuron should be dropped while $m_i = 0$ illustrates that the neuron should be preserved during dropout. To start, we introduce $\boldsymbol{m}_r$ to denote the mask vector of random dropout of the first sub-model. With the dropout ratio $\sigma \in [0, 1]$ preseted, we can define the constraint on $\boldsymbol{m}$ as,

$$\mathcal{R}_{\boldsymbol{m}} = \left\{ \boldsymbol{m} \mid \boldsymbol{m} \in \{0,1\}^N, \|\boldsymbol{m}\|_0 = \lfloor \sigma N \rfloor \right\}, \quad (1)$$

where $\|\cdot\|_0$ is the ℓ_0 norm, N is the number of neurons in the model. And then, we can obtain the worst-case mask vector $\boldsymbol{m}_w$ via maximizing the discrepancy of the outputs from two sub-models sampled with dropout,

$$\boldsymbol{m}_w = \underset{\boldsymbol{m}\in\mathcal{R}_{\boldsymbol{m}}}{\arg\max} \underset{(\mathbf{x},y)\sim\mathcal{D}}{\mathbb{E}} \mathcal{D}_{KL}\left(f_{\boldsymbol{\theta}}(\mathbf{x}, \boldsymbol{m}_r) || f_{\boldsymbol{\theta}}(\mathbf{x}, \boldsymbol{m})\right), \quad (2)$$

where $\mathcal{D}_{KL}$ is Kullback–Leibler divergence. With Taylor expansion, we can approximate the optimal solution as,

$$\boldsymbol{m}_w \approx \underset{\boldsymbol{m}\in\mathcal{R}_{\boldsymbol{m}}}{\arg\max} \frac{1}{2} \boldsymbol{m}^T \boldsymbol{H} \boldsymbol{m}, \quad (3)$$

$$\boldsymbol{H} = \underset{(\mathbf{x},y)\sim\mathcal{D}}{\mathbb{E}} \nabla^2 \mathcal{D}_{KL}\left(f_{\boldsymbol{\theta}}(\mathbf{x}, \boldsymbol{m}_r) || f_{\boldsymbol{\theta}}(\mathbf{x}, \boldsymbol{m})\right) \Big|_{\boldsymbol{m}=\mathbf{0}}, \quad (4)$$

$\boldsymbol{H}$ is the Hessian matrix of the loss $\mathcal{D}_{KL}$ at $\boldsymbol{m} = \mathbf{0}$ which is a $N \times N$ semi-positive definite matrix. Obviously, this is a Binary Quadratic Programming (BQP) problem, which is NP-hard but admits an approximate solution to $\boldsymbol{m}_w$. Here, we introduce a novel and more suitable method for this problem, which enjoys more accurate solution of semidefinite programming (SDP) relaxations [6] and higher efficiency of spectral relaxations methods [7] simultaneously. Firstly, we convert $\{0, 1\}$-constraint on $\boldsymbol{m}$ of $\mathcal{R}_{\boldsymbol{m}}$ to $\{-1, 1\}$-constraint on $\boldsymbol{n}$ via defining $\boldsymbol{n} = 2\boldsymbol{m} - 1$. Then we introduce a new variable $\widehat{\boldsymbol{n}}$ with $N + 1$ dimensions, i.e., $\widehat{\boldsymbol{n}}_i = [\boldsymbol{n}; 1]$, where $[\cdot;\cdot]$ denotes concatenation. We can rewrite constraint $\mathcal{R}_{\boldsymbol{m}}$ as a new constraint $\mathcal{R}_{\widehat{\boldsymbol{n}}}$ on $\widehat{\boldsymbol{n}}$,

$$\mathcal{R}_{\widehat{\boldsymbol{n}}} = \left\{ \widehat{\boldsymbol{n}} \mid \widehat{\boldsymbol{n}} \in \{\pm 1\}^{N+1}, \boldsymbol{e}^T \widehat{\boldsymbol{n}} = c \right\}, \quad (5)$$

where $c = 2\lfloor \sigma N \rfloor - N + 1$ and $\boldsymbol{e} \in \mathbb{R}^{N+1}$ is an all-one vector. By these transformations, we can reformulate the BQP in Eq (4) as a new BQP in terms of $\widehat{\boldsymbol{n}}$, where the constraint term $\widehat{\boldsymbol{n}} \in \{\pm 1\}^{N+1}$ can be rewrited as $\widehat{\boldsymbol{n}}_i^2 = 1$. We then introduce a Lagrange multiplier λ_i for each constraint $\widehat{\boldsymbol{n}}_i^2 = 1$ and λ_0 for the constraint $\boldsymbol{e}^T \widehat{\boldsymbol{n}} = c$. Now, we can formulate the dual problem of the original BQP as,

$$\min_{\boldsymbol{\lambda}, \lambda_0} d\left(\boldsymbol{\lambda}, \lambda_0\right), \quad (6)$$

$$\begin{aligned} d\left(\boldsymbol{\lambda}, \lambda_0\right) &= \max_{\|\widehat{\boldsymbol{n}}\|^2 = N+1} \widehat{\boldsymbol{n}}^T [\boldsymbol{L} + \operatorname{diag}(\boldsymbol{\lambda})] \widehat{\boldsymbol{n}} - \boldsymbol{e}^T \boldsymbol{\lambda} - c\lambda_0 \\ &= (N+1)\lambda_{\max} - \boldsymbol{e}^T \boldsymbol{\lambda} - c\lambda_0, \end{aligned} \quad (7)$$

where

$$\boldsymbol{L} = \begin{pmatrix} \boldsymbol{H} & \boldsymbol{H}\boldsymbol{e} + \frac{1}{2}\lambda_0 \boldsymbol{e} \\ \boldsymbol{e}^T \boldsymbol{H} + \frac{1}{2}\lambda_0 \boldsymbol{e}^T & 0 \end{pmatrix} \in \mathbb{R}^{(N+1)\times(N+1)}$$

, and $\lambda_{\max}$ is the largest eigenvalue of $\boldsymbol{L} + \operatorname{diag}(\boldsymbol{\lambda})$. The eigenvector of unit norm $\mathbf{u}_{\max}$ corresponding to $\lambda_{\max}$ can be derived via approximated by using a single-step power iteration instead of conducting naive eigenvalue decomposition. Then, the maximum $\widehat{\boldsymbol{n}}^*$ can be derived,

$$\widehat{\boldsymbol{n}}^* = \sqrt{N+1}\mathbf{u}_{\max}. \quad (8)$$

The dual problem Eq.(6) can be solved by the gradient descent method. The gradients of d w.r.t $\boldsymbol{\lambda}$ and λ_0 are as follows,

$$\nabla_{\boldsymbol{\lambda}} d = (N+1)\mathbf{u}_{\max}^2 - \boldsymbol{e}, \quad (9)$$

$$\frac{\partial d}{\partial \lambda_0} = \frac{1}{2}(N+1)\mathbf{u}_{\max}^T \begin{pmatrix} 0 & \boldsymbol{e} \\ \boldsymbol{e} & 0 \end{pmatrix} \mathbf{u}_{\max} - c, \quad (10)$$

where $\mathbf{u}_{\max}^2$ denotes an element-wise square of $\mathbf{u}_{\max}$. During training, over each mini-batch, we compute the above gradient to make a one-step update of the Lagrange multipliers $\boldsymbol{\lambda}$ and λ_0 with the gradients, before the maximum $\widehat{\boldsymbol{n}}^*$ is taken with the updated multipliers. Finally, both $\pm\widehat{\boldsymbol{n}}^*$ are optimal and we should choose the one closer to $\widehat{\boldsymbol{n}}_{N+1} = 1$ as required. Note that we seek the worst-case dropout mask layer-by-layer instead of applying it to an entire network as a whole. This can make the WordReg computationally efficient as well as prevent too many neurons from being dropped at a few layers. Given the small scale of neurons (usually $<1k$) in each layer in widely-used DNNs, WordReg is computationally cheap. With the worst-case dropout mask vector $\boldsymbol{m}_w$ obtained, we can formulate the loss $\mathcal{L}$ of DNNs training with WordReg as,

$$\mathcal{L} = \underset{(\mathbf{x},y)\sim\mathcal{D}}{\mathbb{E}} \left(l\left(y, f_{\boldsymbol{\theta}}\left(\mathbf{x}\right)\right) + \mu \mathcal{D}_{KL}\left(f_{\boldsymbol{\theta}}(\mathbf{x}, \boldsymbol{m}_r), f_{\boldsymbol{\theta}}(\mathbf{x}, \boldsymbol{m}_w)\right) \right), \quad (11)$$

where $l(\cdot)$ is the loss of neural network training and μ is a trade-off parameter that we will study in section 4.4.

3. THEORETICAL JUSTIFICATION

In this section, we aim to explain the reasons why WordReg can mitigate the gap between training and inference. Specifically, we show that the inconsistency between loss of the full model $f_{\boldsymbol{\theta}}(\mathbf{x})$ and the averaged loss of sub-models can be bounded by our regularization objective using a linear model.

Theorem 3.1. *Given the liner model is $f_{\boldsymbol{\theta}}(\mathbf{x}) = softmax$ $(Norm(\mathbf{w}^T\mathbf{x}))$ where $Norm(\cdot)$ is the normalization layer, we have:*

$$
\begin{aligned}
&|\mathcal{L}(w) - \mathbb{E}_{\boldsymbol{m}_r}[\mathcal{L}(w, \boldsymbol{m}_r)]| \\
&\leq c\sqrt{\mathop{\mathbb{E}}_{(\mathbf{x},y)\sim\mathcal{D}} \mathcal{D}_{KL}\left(f_{\boldsymbol{\theta}}(\mathbf{x}, \boldsymbol{m}_r) || f_{\boldsymbol{\theta}}(\mathbf{x}, \boldsymbol{m}_w)\right)},
\end{aligned}
\tag{12}
$$

where c is a constant.

Proof. Given the loss function $\mathcal{L}$ is c_1-Lipschitz smooth, we have,

$$
\begin{aligned}
&\frac{1}{c_1}|\mathcal{L}(w) - \mathbb{E}_{\boldsymbol{m}_r}[\mathcal{L}(w, \boldsymbol{m}_r)]| \leq \\
&\mathop{\mathbb{E}}_{\substack{\boldsymbol{m}_r,\\ (\mathbf{x},y)\sim\mathcal{D}}} \left\| \mathbf{w}^T\mathbf{x} - \frac{\left(\mathbf{w}^T\mathbf{x}\right) \odot \boldsymbol{m}_r}{\sigma} \right\| = \mathop{\mathbb{E}}_{(\mathbf{x},y)\sim\mathcal{D}} (1-\sigma)\left\|\mathbf{w}^T\mathbf{x}\right\|,
\end{aligned}
\tag{13}
$$

where σ is the dropout ratio and is the Lipshitz constant. Based on the relation between the KL-divergence and the total variation distance, we have,

$$
\begin{aligned}
&\mathop{\mathbb{E}}_{\substack{\boldsymbol{m}_r,\boldsymbol{m}_r'\\ (\mathbf{x},y)\sim\mathcal{D}}} \|f_{\boldsymbol{\theta}}(\mathbf{x}, \boldsymbol{m}_r) - f_{\boldsymbol{\theta}}(\mathbf{x}, \boldsymbol{m}_r')\|_1 \\
&\leq \sqrt{2 \mathop{\mathbb{E}}_{\substack{\boldsymbol{m}_r,\boldsymbol{m}_r'\\ (\mathbf{x},y)\sim\mathcal{D}}} \mathcal{D}_{KL}\left(f_{\boldsymbol{\theta}}(\mathbf{x}, \boldsymbol{m}_r) || f_{\boldsymbol{\theta}}(\mathbf{x}, \boldsymbol{m}_r')\right)} \\
&\leq \sqrt{2 \mathop{\mathbb{E}}_{\substack{\boldsymbol{m}_r\\ (\mathbf{x},y)\sim\mathcal{D}}} \mathcal{D}_{KL}\left(f_{\boldsymbol{\theta}}(\mathbf{x}, \boldsymbol{m}_r) || f_{\boldsymbol{\theta}}(\mathbf{x}, \boldsymbol{m}_w)\right)},
\end{aligned}
\tag{14}
$$

Suppose that the inverse function from $softmax(\mathbf{w}^T\mathbf{x})$ to $\mathbf{w}^T\mathbf{x}$ is c_2-Lipschitz smooth, we have,

$$
\begin{aligned}
&\mathop{\mathbb{E}}_{\substack{\boldsymbol{m}_r,\boldsymbol{m}_r'\\ (\mathbf{x},y)\sim\mathcal{D}}} \left\| \frac{\left(\mathbf{w}^T\mathbf{x}\right) \odot \boldsymbol{m}_r - \left(\mathbf{w}^T\mathbf{x}\right) \odot \boldsymbol{m}_r'}{\sigma} \right\| \\
&= 2(1-\sigma) \mathop{\mathbb{E}}_{(\mathbf{x},y)\sim\mathcal{D}} \left\|\mathbf{w}^T\mathbf{x}\right\| \\
&\leq c_2\sqrt{2 \mathop{\mathbb{E}}_{\substack{\boldsymbol{m}_r\\ (\mathbf{x},y)\sim\mathcal{D}}} \mathcal{D}_{KL}\left(f_{\boldsymbol{\theta}}(\mathbf{x}, \boldsymbol{m}_r) || f_{\boldsymbol{\theta}}(\mathbf{x}, \boldsymbol{m}_w)\right)},
\end{aligned}
\tag{15}
$$

because both $\boldsymbol{m}_r, \boldsymbol{m}_r'$ independently follow Bernoulli distribution. Unifying Eq.(13) and Eq.(15), we have,

$$
\begin{aligned}
&|\mathcal{L}(w) - \mathbb{E}_{\boldsymbol{m}_r}[\mathcal{L}(w, \boldsymbol{m}_r)]| \\
&\leq \frac{\sqrt{2}}{2}c_1c_2\sqrt{\mathop{\mathbb{E}}_{(\mathbf{x},y)\sim\mathcal{D}} \mathcal{D}_{KL}\left(f_{\boldsymbol{\theta}}(\mathbf{x}, \boldsymbol{m}_r) || f_{\boldsymbol{\theta}}(\mathbf{x}, \boldsymbol{m}_w)\right)},
\end{aligned}
\tag{16}
$$

Let $c = \frac{\sqrt{2}}{2}c_1c_2$, we can draw the conclusion. □

4. EXPERIMENTS

In this section, we experimentally verify that WordReg can be universally applied in diverse DNNs including Convolutional Neural Networks (CNNs), Transformer, Graph Neural Networks (GNNs), and diverse tasks including image recognition, language understanding, and graph classification.

4.1. Application to Image Recognition

Table 1. Accuracy on CIFAR-100 with 3 random seeds while ImageNet with only 1 seed for the heavy computational overhead. Kindly note that ResNet-34 here is trained from scratch while the ViT models are initialized with publicly available pre-trained weights and then finetuned on the CIFAR-100 and ImageNet datasets.

Method	CIFAR-100	ImageNet
ResNet-34 [8]	78.59 ± 0.16	75.14
ResNet-34 + FD [5]	78.94 ± 0.10	75.54
ResNet-34 + WordReg	**79.85** ± 0.24	**76.41**
ViT-B/16 [9]	92.55 ± 0.36	83.86
ViT-B/16 + FD [5]	92.76 ± 0.31	84.03
ViT-B/16 + WordReg	**93.92** ± 0.22	**84.54**
ViT-L/16 [9]	93.41 ± 0.27	85.12
ViT-L/16 + FD [5]	93.73 ± 0.16	85.36
ViT-L/16 + WordReg	**94.75** ± 0.19	**86.05**

For image recognition, we perform experiments on two representative datasets: CIFAR-100 [14] and ImageNet [15]. The CIFAR-100 dataset consists of 60k images from 100 classes, and each class contains 600 images with 500 for training and 100 for testing. The ImageNet dataset contains 1.3M images from 1k classes. We adopt ResNet [8] and recent Vision Transformer [9] (ViT) as backbones. The data preprocessing strategies and other details are the same as the two original works, respectively. Additionally, we set the hyperparameter μ as 0.60 in this experiment. We show the results in Table 1, from which we can observe that WordReg brings $\sim 1.3\%$ accuracy improvements across ResNet-34 and ViT models on CIFAR-100 datasets. Similar achievements can also be observed on the large-scale dataset ImageNet. These verify that WordReg can benefit various DNNs in the task of image recognition.

4.2. Application to Language Understanding

Pre-trained Language Models (PLMs) such as BERT [10] have achieved remarkable success in the task of language understanding. Considering that pre-training such large-scale language models from scratch are expensive, we only evaluate WordReg in the fine-tuning stage. Specifically, we fine-tune two publicly available PLMs: BERT-base [10] and RoBERTa-large [13] on the GLUE [16] benchmark, which contains 8 text classification or regression tasks. Note that we substitute KL divergence with MSE in the regression task (STS-B). For each task, we tune the hyper-parameter μ in the set $\{0.2, 0.4, 0.6\}$. The evaluation metrics for the above 8 tasks are as follows: The result for STS-B is the Pearson correlation; Matthew's correlation is used for CoLA; Other tasks are measured by

Table 2. Fine-tuned model performances on GLUE language understanding benchmark. The results on the development set are a median over five runs. Kindly note that it is a common practice not to show the standard deviation on this benchmark.

Model	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	CoLA	Avg.
BERT-base [10]	83.8	85.3	90.8	91.0	68.2	92.4	89.3	62.3	82.85
BERT-base + WordReg	85.7	87.8	92.6	91.9	71.7	93.5	90.1	63.2	**84.56**
XLNet-large [11]	90.8	90.8	94.9	92.3	85.9	97.0	92.5	69.0	89.15
ELECRTA-large [12]	90.9	90.8	95.0	92.4	88.0	96.9	92.6	69.1	89.46
RoBERTa-large [13]	90.2	90.9	94.7	92.2	86.6	96.4	92.4	68.0	88.93
RoBERTa-large + WordReg	91.0	91.9	95.8	92.9	89.0	97.4	92.7	70.7	**90.18**

Table 3. Accuracy (%) on graph classification tasks. We show the mean and standard deviation over 10 different runs.

Datasets	MUTAG	PROTEINS	D&D
GCN [18]	69.50±1.78	73.24±0.73	72.05±0.55
GCN + WordReg	71.89±1.01	74.76±0.64	74.14±0.87
GIN [19]	81.39±1.53	71.46±1.66	70.79±1.17
GIN + WordReg	82.45±1.37	72.95±0.75	72.21±0.97

Accuracy. Additionally, we keep other experimental settings as previous works [10, 13]. The results in Table 2 indicate that fine-tuning PLMs with WordReg consistently outperforms vanilla fine-tuning by a notable margin, which further verifies the effectiveness of WordReg. Also, RoBERTa-large + WordReg achieves superior performance over the state-of-the-art (SOTA) PLMs including XLNet-large [11] and ELECTRA-large [12] in the tasks of language understanding.

4.3. Application to Graph Classification

Graph classification aims to label a given graph with the maximum probability among several seed categories. We use 3 benchmarks for graph classification from TU datasets [17], which include MUTAG, PROTEINS, D&D. We employ GCN [18] and GIN [19] as the base models. Additionally, we evaluate the model performance with a 10-fold cross-validation setting, where the dataset split is based on the conventionally used training/test splits [20]. We use the early stopping criterion, where we stop the training if there is no further improvement on the validation loss during 50 epochs. Furthermore, the maximum number of epochs is set to 500. We then report the average performances on test sets, by performing overall experiments 10 times. The results shown in Table 3 reveal that GNNs with WordReg achieve superior performance over vanilla GNNs training, which further validates the effectiveness of WordReg in various applications.

4.4. Hyper-parameters Sensitivity Analysis

We substitute the worst-case dropout with random dropout to study its influence. As shown in Figure 2, WordReg outperforms the random dropout across various drop ratios, which validates that searching for the worst-case dropout is necessary and conducive. Also, we can make the following observations: (1) WordReg can achieve better performance when the two dropout ratios are in a reasonable range (0.2-0.4). (2) WordReg tends to perform better when the two dropout ratios are close

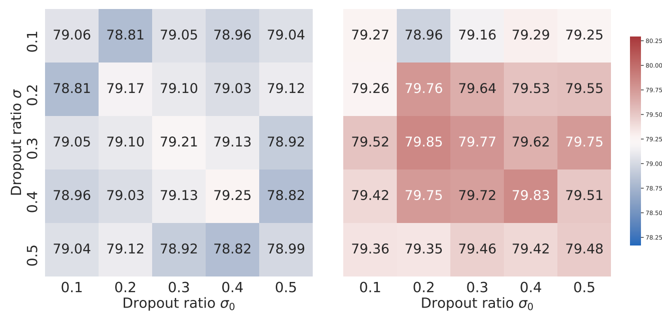

Fig. 2. Comparisons between WordReg (right) and random dropout regularization (left). The experiments are conducted on CIFAR-100 with ResNet-34. σ_0 and σ are dropout ratios of vanilla drop and worst-case drop, respectively. The left subfigure is symmetrical because both two dropouts are random.

Table 4. The influence of the trade-off parameter μ on CIFAR-100 (ResNet-34) and GLUE benchmark (BERT-base).

μ	0.0	0.2	0.4	0.6	0.8	1.0
CIFAR-100	78.59	79.52	**79.85**	79.65	79.62	79.13
GLUE Benchmark	82.85	83.91	84.24	**84.56**	84.32	83.83

or identical. Additionally, we study the trade-off parameter μ in Table 4. Initially, the prediction performance will be improved with the increase of μ. However, the performance sees a dramatic drop after a specific threshold, which can be attributed to its over-powerful regularization.

5. CONCLUSION

In this paper, we propose a simple yet effective regularization, dubbed WordReg, to mitigate the gaps between training and inference. Specifically, we formulate the process of searching for worst-case dropout as a BQP problem and introduce a novel and more suitable method to obtain the approximate solution. Extensive experiments verify that WordReg is universally effective in various scenarios. Due to the limited computational resources, we have not tested WordReg in the pre-training stage of BERT, which we leave as future works.

6. ACKNOWLEDGMENTS

This work is supported by the Science and Technology Innovation 2030 - Major Project (No. 2021ZD0150100) and National Natural Science Foundation of China (No. U21A20427).

7. REFERENCES

[1] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[2] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus, "Regularization of neural networks using dropconnect," in *International conference on machine learning*. PMLR, 2013, pp. 1058–1066.

[3] Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. PMLR, 2015, pp. 448–456.

[4] Xuezhe Ma, Yingkai Gao, Zhiting Hu, Yaoliang Yu, Yuntian Deng, and Eduard H. Hovy, "Dropout with expectation-linear regularization," *ICLR*, vol. abs/1609.08017, 2017.

[5] Konrad Zolna, Devansh Arpit, and others., "Fraternal dropout," in *ICLR*, 2018.

[6] Qing Zhao, Stefan E Karisch, Franz Rendl, and Henry Wolkowicz, "Semidefinite programming relaxations for the quadratic assignment problem," *Journal of Combinatorial Optimization*, vol. 2, no. 1, pp. 71–109, 1998.

[7] Carl Olsson, Anders P Eriksson, and Fredrik Kahl, "Solving large scale binary quadratic problems: Spectral methods vs. semidefinite programming," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2007, pp. 1–8.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021.

[10] Jacob Devlin, Ming-Wei Chang, and others., "Bert: Pre-training of deep bidirectional transformers for language understanding," *NAACL*, 2019.

[11] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le, "Xlnet: Generalized autoregressive pretraining for language understanding," *Advances in neural information processing systems*, vol. 32, 2019.

[12] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning, "ELECTRA: Pre-training text encoders as discriminators rather than generators," in *ICLR*, 2020.

[13] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[14] Alex Krizhevsky, Geoffrey Hinton, et al., "Learning multiple layers of features from tiny images," 2009.

[15] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[16] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman, "GLUE: A multi-task benchmark and analysis platform for natural language understanding," in *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, Brussels, Belgium, Nov. 2018, pp. 353–355, Association for Computational Linguistics.

[17] Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann, "Tudataset: A collection of benchmark datasets for learning with graphs," in *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*.

[18] N. Thomas Kipf. and M. Welling, "Semi-supervised classification with graph convolutional networks," *ICLR*, 2017.

[19] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka, "How powerful are graph neural networks?," *ICLR*, 2018.

[20] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen, "An end-to-end deep learning architecture for graph classification," in *Proceedings of the AAAI conference on artificial intelligence*, 2018, vol. 32.